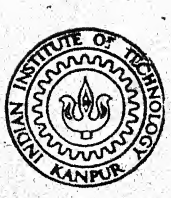
TWO-DIMENSIONAL CUTTING STOCK PROBLEM

By

SURESH CHAND



DEPARTMENT OF MECHANICAL ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY KANPUR

AUGUST 1974

TWO-DIMENSIONAL CUTTING STOCK PROBLEM

A Thesis Submitted
In Partial Fulfilment of the Requirements
for the Degree of
MASTER OF TECHNOLOGY

By
SURESH CHAND

to the

DEPARTMENT OF MECHANICAL ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY KANPUR
AUGUST 1974

CERTIFICATE

Certified that present work on 'Two-Dimensional Cutting Stock Problem' by Suresh Chand has been carried out under my supervision and has not been submitted elsewhere for the award of a degree.

(S.C. Nayak)
Assistant Professor
Department of Mechanical Engineering
Indian Institute of Technology, Kanpur

ACKNOWLEDGEMENT

I wish to express my deep sense of gratitude to Dr. S.C.Nayak for introducing the present problem to me and for his constant encouragement and invaluable guidance throughout the completion of this work.

I am thankful to Mr. S.S. Sibia for his useful suggestions from time to time.

Last , but not the least, I thank Mr. D.P. Saini for his efficient typing of the manuscript.

Suresh Chand

CONTENTS

LIST OF TABLES

LIST OF FIGURES

SYNOPSIS

Cutting stock problem arises in various industries where stock sizes are required to be cut into order sizes. The management is interested in finding the cutting patterns that will fill an order at minimum cost. Present work deals with the problems where order sizes and stock sizes are rectangular.

In the past, work has been done on this problem for guillotine cutting case, and Gilmore and Gomory have suggested a method which gives optimal results. But this method of cutting gives large wastage because it permits only limited flexibility for the combinations of demand rectangles to be cut from a stock plate. This thesis considers the problem where cutting equipment permits the cutting of all possible combinations of demand rectangles from a stock plate. Such a problem commonly arises in the cutting of thick plates by gas cutting. An attempt has been made to develop an heuristic algorithm to solve this problem. This algorithm considers all the cutting patterns permitted in two stage guillotine cutting. Along with these, some more cutting patterns are considered which are generated by the heuristic algorithm.

This method requires solving a number of knapsack problems. An efficient branch and search program has been developed for this.

It is found that this method gives considerable saving in scrap as compared to the two stage guillotine cutting method suggested by Gilmore and Gomory. But this saving is at the cost of large computer time.

CHAPTER 1

INTRODUCTION

A problem of primary significance to a variety of industries is the reduction of trim losses in cutting of materials for the execution of business orders. Typically, this problem is found in paper, glass, and steel industries. Management receives orders for various shapes and sizes of material. These orders are filled from larger sizes available in stock. It is generally found that trimming losses due to odd pieces are unavoidable. The management is interested in generating cutting patterns that will fill an order at minimum cost. This is known as cutting stock problem in the field of Operations Research.

In most of the industries, the order sizes are either one or two-dimensional. For example, the order may consist of rolls of given widths of a material or rectangular pieces of given widths and lengths etc. Correspondingly, there are two types of cutting stock problems.

One-dimensional cutting stock problem arises in cutting of rolls of paper, textile industry, cutting of sections or channels, bars, squares, flats, etc. in merchant mills of steel plants. The problem is to minimize the scrap losses. Gilmore and Gomory[1,2] suggested a solution to this problem which gives optimal results.

Two-dimensional cutting stock problem arises when there is demand for two dimensional shapes like rectangles, circles etc. The stock sizes are generally rectangular. There are different variations of this problem depending on the production process.

Thin sheets are generally cut by two stage guillotine cutting. This problem can be treated as equivalent to two one-dimensional cutting stock problems and optimal results can be found by the method suggested by Gilmore and Gomory[3].

In glass industry, the process is continuous and defective glass plates are to be cut into demand sizes. The defective plates are coming out of a production shop and they are to be cut in the same sequence in which they are coming out. The cutting equipment is capable of guillotine cutting only. The sizes of the cuts can be varied. Susan. G. Hahn[4] has suggested a solution to such a problem. The optimality of the results can not be guaranteed because some subjective decisions have to be made.

Two-dimensional cutting stock problem is of special significance to heavy engineering industry, where thick plates of scarce material are required to be cut into smaller order sizes. Generally gas cutting is used for the cutting of thick plates. The concept of "two stage guillotine cutting" can be used for thick plates also, and optimal solution for this is available. The draw back of this method of cutting is that it permits less flexibility for the combinations of order sizes to be cut from a stock rectangle, thus resulting in more scrap losses. So there

is a possibility of reduction in wastage if an algorithm is developed which permits more flexibility.

At present, this problem is usually solved by hand in industries. Experienced men regularly achieve good cutting patterns. But this method is time consuming and depends on human judgement. Also, it is difficult even for an experienced operator to select the "best" combination without actually trying some arrangements.

In this work, an attempt has been made to develop an heuristic algorithm which provides more flexibility for the combinations of order sizes to be cut from a stock rectangle and also gives the layout of the cutting.

In chapter 2, literature on one and two-dimensional cutting stock problems and knapsack problem has been reviewed.

Chapter 3, gives the development of the heuristic algorithm for solving two-dimensional cutting stock problem. An efficient branch and search program for solving the knapsack problem has also been developed.

Results for some randomly generated problems are reported in chapter 4. A comparison has been made with the results obtained for the same problems by the method suggested by Gilmore and Gomory for two stage guillotine cutting case.

CHAPTER 2

LITERATURE REVIEW

2.1 One-Dimensional Cutting Stock Problem:

The simplest kind of cutting stock problem is the one-dimensional cutting stock problem. Consider that k standard lengths $L_1, L_2, \ldots, L_k$ of a material are available having costs $C_1', C_2', \ldots, C_k'$ respectively. These will be termed as stock lengths. It is assumed that a large number of each of the stock lengths is available. An order has been received for lengths $l_1, l_2, \ldots, l_m$ of the material. These lengths will be termed as demand lengths. Let the units of length l_i demanded be N_i. It is required to find a cutting arrangement to cut the demand lengths from the stock lengths such that the total order cost is minimized.

This problem was first formulated by Kurt Eisman[5] as a linear programming problem. He suggested that all the possible cutting patterns should be enumerated. A cutting pattern shows the number of different demand sizes that can be cut from a stock size. It can be written as a column vector having m components. Let a_{ij} be the i^{th} element of j^{th} column. This represents the number of demand lengths l_i in j^{th} cutting pattern. Let a cutting pattern be called as an activity. The total number of activities will depend on the problem. Now the problem becomes:

which activities should be chosen and how many times an activity should be repeated so that the requirements for demand lengths are satisfied at minimum cost? It can be expressed as a linear integer programming problem as follows:

$$\text{Minimize} \quad \sum_{j=1}^{n} C_j x_j$$

$$\text{such that} \quad \sum_{j=1}^{n} a_{ij} x_j \geq N_i , \quad i = 1, 2, \ldots, m$$

$$x_j \geq 0 , \quad x_j \text{ is an integer}, \quad j = 1, 2, \ldots, n$$

where n is the total number of activities and x_j is the number of times j^{th} activity is repeated. C_j is the cost of the stock length cut by j^{th} activity.

Due to unavailability of an efficient method for solving integer programming problems, Kurt Eisman[5] suggested that x_j's can be treated as continuous variables so that the problem can be solved as a simple linear programming problem. The x_j's so obtained will in general be fractional. They can be rounded to nearest smaller integers. It will leave some of the orders unfilled.

Gilmore and Gomory[1,3] did a detailed study of this problem. If the integer restriction on the variables is dropped, the problem can be expressed as:

$$\text{Minimize} \quad \sum_{j=1}^{n} C_j x_j$$

such that $\sum_{j=1}^{n} a_{ij} x_j - x_{n+i} = N_i$, $i = 1,2,\ldots,m$

$$x_j \geq 0 \ , \ j = 1,2,\ldots,n$$

where $x_{n+1}, x_{n+2}, \ldots, x_{n+m}$ are surplus variables.

It has been proved * that if there is a solution to this problem in which all the surplus variables are not zeroes, then there exists a solution with the same cost in which all the surplus variables are zeroes. So the surplus variables can be dropped and the problem can be simplified as:

Minimize $\sum_{j=1}^{n} C_j x_j$

such that $\sum_{j=1}^{n} a_{ij} x_j = N_i$, $i = 1,2,\ldots,m$

$$x_j \geq 0 \ , \quad j = 1,2,\ldots,n$$

Simplex method can be used to solve this problem. The main disadvantage of such a formulation is that even for a small problem, the number of columns (n) is too large for a modern computer.

In order to overcome this difficulty, Gilmore and Gomory[1,3] suggested a column generation technique. It is an efficient way of searching the best column. The best column is found by solving an auxiliary problem which is a knapsack problem.

Let B be the current basis matrix containing m columns $[Q_1, Q_2, \ldots, Q_m]$. Column Q_i is a vector containing m elements. Let Q be a nonbasic activity $a_1, a_2, \ldots, a_m$ with cost C. The vector Q

---

* NOTE: The proof for this has been given by Gilmore and Gomory[1]. This propertyholds for this problem because of it's special structure.

will give reduction in cost in current iteration if and only if:

$$C_B B^{-1} Q > C \quad \ldots\ldots \quad (2.1)^*$$

where C_B is a m component row vector consisting of costs corresponding to basic variables. Let $C_B B^{-1} = [v_1, v_2, \ldots\ldots, v_m]$ then (2.1) can be written as:

$$v_1 a_1 + v_2 a_2 + \ldots\ldots + v_m a_m > C$$

So a column vector Q is to be chosen such that

$$v_1 a_1 + v_2 a_2 + \ldots\ldots + v_m a_m > C \quad \ldots\ldots \quad (2.2)$$

and $$l_1 a_1 + l_2 a_2 + \ldots\ldots + l_m a_m \leq L_K, \; K \in [1,2,\ldots,k] \quad \ldots\ldots \quad (2.3)$$

Such a column vector Q can be found by solving following knapsack problem

$$\text{Maximize} \quad \sum_{i=1}^{m} v_i y_i \quad \ldots\ldots \quad (2.4)$$

$$\text{such that} \quad \sum_{i=1}^{m} l_i y_i \leq L_K, \; K \in [1, 2, \ldots, k] \quad (2.5)$$

y_i is a positive integer

Let $[y_1^*, y_2^*, \ldots, y_m^*]$ be a solution to this problem. If this solution does not satisfy conditions (2.2) and (2.3) then no other columns from the original problem will satisfy these conditions. If v_i is interprated as value of i^{th} demand piece, then the objective of this

---

* Note: In linear programming, for a minimization problem, a column A_j is a candidate for entering into the basis if and only if $z_j - C_j > 0$. Putting $z_j = C_B B^{-1} A_j$, this condition becomes $C_B B^{-1} A_j > C_j$, which is same as condition (2.1).

knapsack problem is to choose a cutting pattern such that the total value of demand sizes in the cutting pattern is maximized. This approach makes the linear programming formulation usable for problems of reasonable sizes.

2.2 Two-Dimensional Cutting Stock Problem:

2.2.1 Two Stage Guillotine Cutting:

In this problem, a large number of stock rectangles, each of width W and length L is available. There is an order for amount N_i of a demand rectangle of width w_i and length l_i. It is required to find a cutting arrangement to cut demand rectangles from the stock rectangles by using minimum number of stock rectangles. It can be formulated as a linear programming problem as done in one-dimensional case:

$$\text{Minimize} \quad \sum_{j=1}^{n} x_j \qquad \ldots \qquad \ldots \qquad (2.6)$$

$$\text{such that} \quad \sum_{j=1}^{n} a_{ij}\, x_j = N_i \quad , \; i = 1,2,\ldots,m \qquad (2.7)$$

$$x_j \geq 0$$

where a_{ij} is the number of i^{th} demand rectangles in j^{th} cutting pattern and x_j is the number of times j^{th} activity (cutting pattern) is to be repeated. n is the total number of cutting patterns.

The auxiliary problem for column generation is:

$$\text{Maximize} \quad \sum_{i=1}^{m} v_i\, y_i \qquad \ldots \qquad \ldots \qquad (2.8)$$

$$\text{such that} \quad \left[y_1 , y_2, \ldots\ldots, y_m\right] \text{ is a feasible cutting pattern} \qquad \ldots \qquad (2.9)$$

where m, y_i and v_i have the same meaning as discussed earlier.

It was easy to write (2.9) in mathematical form in case of one-dimensional cutting stock problem but the same is not true for two-dimensional case.

Gilmore and Gomory[3] have suggested a solution to the auxiliary problem with the assumption of two stage guillotine cutting. In it, the plate is first cut into strips along the length of the plate. The width of a strip is equal to the width of atleast one of the demand rectangles. These strips are in turn used for cutting the demand rectangles. The problem of finding a cutting pattern for a stock rectangle such that total value is maximized can be treated as equivalent to following two problems in one-dimensional case:

(i) Choosing a cutting pattern for a strip of known width which gives maximum value of the strip.

(ii) Finding out a combination of strips to be cut from the stock plate such that total value is maximized.

2.2.2 Cutting of Defective Plates:

Consider now the cutting of defective plates. In a continuous production process, some of the plates produced may be defective. Such a problem commonly arises in glass industries. Susan G. Hahn[5] has suggested a solution to this problem. It has been assumed that the cutting equipment allows only guillotine cutting. The delivery dates for the demand rectangles are given.

It is obvious that pieces from a stock plate should be cut according to following policies:

(i) Preference should be given to the demand sizes having early delivery dates and higher demands.

(ii) Preference should be given to the larger area pieces as compared to smaller area pieces, because it is more difficult to fit larger pieces. At the same time, care should be taken of the fact that cutting of a larger area piece from a more defective plate gives more wastage.

Hahn[5] suggests that values should be assigned to the demand rectangles based on areas, due dates etc., and dynamic programming approach can be used to solve the problem.

2.2.3 Generalized Two-Dimensional Cutting Stock Problem:

So far, it has been assumed that cutting equipment permits only guillotine cutting and demand sizes are rectangular. In the generalized version of the problem, the demand sizes can be in any shapes and the cutting equipment permits the cutting of all possible combinations of demand sizes. Adamowicz and Albano[6] have suggested a two stage algorithm to solve this problem. It has been claimed by these authors that the results may not be optimal but generally they are good.

The first stage of the algorithm is the clustering stage. It produces rectangular enclosures for the shapes or for various clusterings of shapes according to certain policies. It also gives the numbers

of different enclosures to be cut. In the second stage, these authors have suggested an heuristic algorithm to find the cutting pattern for a stock rectangle such that trim losses for the stock plate are minimized without overproducing any demand rectangle.

The draw back of this method is that while it attempts to minimize the trim losses for individual plates, it does not consider the effect of one cutting pattern on another. Secondly, this method is not suitable for problems having high demands for sizes.

2.2.4 Present Study:

In this work, an attempt has been made to develop an heuristic algorithm for the generalized version of the two-dimensional cutting stock problem. It has been assumed that demand sizes are rectangular. This algorithm takes into consideration all the cutting patterns permitted in two stage guillotine cutting. These cutting patterns are slightly modified as discussed in chapter 3. In addition to these, it considers some more cutting patterns generated by the heuristic algorithm.

2.3 Solution of Knapsack Problem:

The efficiency of the proposed method depends on the efficiency with which knapsack problem could be solved.

Knapsack problem has been of interest to research workers because it arises in various situations like - capital budgeting, network reliability and capital investment etc. It often occurs as a subproblem in a problem of optimization.

Gilmore and Gomory[7] suggested an efficient dynamic programming approach for solving this problem. This method takes into account the periodicity of the solution. The draw back of this approach is that it requires a large amount of computer memory.

Greenberg[8] suggested a branch and search technique when each variable y_i can take only two values 0 or 1. This method can also be applied to integer solution. There will be only a finite number of solutions so a method based on successive partitioning can be used. The branch and search algorithm proceeds by repeatedly partitioning a class of feasible solutions into smaller and smaller subclasses along a branch till the lower bound is reached or a new feasible solution is found. It then back tracks and new branches of the tree are developed. Further branching from a node is excluded when the lower bound is reached. Algorithm stops when all new branches are excluded.

It requires solving following linear programming problem at each node:

$$\text{Maximize} \quad \sum_{i=1}^{m} v_i y_i$$

$$\text{such that} \quad \sum_{i=1}^{m} l_i y_i \leq L$$

$$0 \leq y_i \leq 1 , \quad i = 1,2,\ldots\ldots,m$$

$$y_i = Z_i^* , \quad i \in \{I_1\}$$

where $\{I_1\}$ gives the indices of the variables which have been assigned values at the node. Z_i^* is the value of assigned variable y_i at the node.

The variables are first arranged according to decreasing magnitude of v_i/l_i . Let the subscripts be reordered such that $v_1/l_1 \geq v_2/l_2 \geq \cdots\cdots \geq v_m/l_m$. Then the solution to this problem is:

$$\left.\begin{aligned} y_i &= 1 \quad \text{if} \quad i < r \\ y_i &= 0 \quad \text{if} \quad i > r \end{aligned}\right\} \quad i \notin \{I_1\}, \quad r \notin \{I_1\}$$

$$y_i = Z_i^* \quad , \quad i \in \{I_1\}$$

$$y_r = (L - \sum_{i<r} l_i - \sum_{i \in \{I_1\}} Z_i^* \; l_i)/l_r$$

The value of the objective function obtained here gives the upper bound on all the solutions represented by this node. Two branches can be developed from this node, one by assigning $y_r = 0$ and the other by assigning $y_r = 1$ along with the values assigned at the node.

The advantage of this method is that the memory space required is not large. This method has been applied to the original knapsack problem (2.4, 2.5) in chapter 3.

CHAPTER 3

TWO-DIMENSIONAL CUTTING STOCK PROBLEM-GENERALIZED CASE

This chapter considers the solution of two-dimensional cutting stock problem, when the cutting equipment permits the cutting of all possible layouts of demand rectangles in a stock rectangle. The method developed here is an extension of the two stage guillotine cutting method of Gilmore and Gomory. The linear programming formulation of the problem as suggested by Gilmore and Gomory is :

$$\text{Minimize} \quad \sum_{j=1}^{n} x_j$$

$$\text{such that} \quad \sum_{j=1}^{n} a_{ij}\, x_j = N_i \quad , i = 1, 2, \ldots, m$$

$$x_j \geq 0$$

Since the number of columns is going to be too large for a problem of reasonable size, Gilmore and Gomory suggested a column generation technique for choosing the best column. It can be found by solving the following auxiliary problem:

$$\text{maximize} \quad \sum_{i=1}^{m} v_i\, y_i \qquad \ldots \qquad \ldots \qquad (3.1)$$

$$\text{such that} \quad [y_1, y_2, \ldots, y_m] \quad \text{is a feasible cutting pattern} \qquad (3.2)$$

It will be called as problem P.

The objective of this auxiliary problem is to choose a cutting pattern such that the total value of the demand rectangles in the cutting pattern is maximized. The method suggested by Gilmore and Gomory for solving this auxiliary problem considers only the cutting patterns permitted in two stage guillotine cutting.

An heuristic algorithm is proposed for solving the auxiliary problem in the generalized cutting case. It will not be possible to consider all the cutting patterns by this heuristic algorithm also. The optimality of the solution will depend on the total number of cutting patterns considered. Larger is the number of cutting patterns considered, better the results are expected to be. It is possible that some of the cutting patterns permitted in two stage guillotine cutting are not generated by the heuristic algorithm and vice versa. So to increase the total number of columns considered, the column generation method for two stage guillotine cutting and the heuristic algorithm for column generation will be combined in the final algorithm. The column generation method as suggested by Gilmore and Gomory will be slightly modified.

3.1 Heuristic Algorithm for Column Generation:

The difficulty in solving the problem P is due to the feasibility constraint (3.2). One of the possible ways of solving this problem could be by writing (3.2) in mathematical form, but it is not easy to do so. Moreover, the aim here is not just to generate a cutting pattern but also to know it's layout in the stock rectangle. The

heuristic algorithm also gives information about it's layout in the stock rectangle.

Since the total area of the pieces in a cutting pattern should not exceed the area of the stock plate, the auxiliary problem can be written as follows:

$$\text{Maximize} \quad \sum_{i=1}^{m} v_i y_i \qquad \ldots\ldots \qquad (3.3)$$

$$\text{such that} \quad \sum_{i=1}^{m} A_i y_i \leq A \qquad \ldots\ldots \qquad (3.4)$$

$$\text{and} \quad [y_1, y_2, \ldots\ldots, y_m] \text{ corresponds to a feasible cutting pattern} \qquad (3.5)$$

where A_i is the area of i^{th} demand rectangle and A is the total area of the stock plate. y_i is an integer which gives the total number of i^{th} demand rectangles in the cutting pattern. Consider the problem (3.3) and (3.4) without the constraint of feasible cutting pattern:

$$\text{Maximize} \quad \sum_{i=1}^{m} v_i y_i$$

$$\text{such that} \quad \sum_{i=1}^{m} A_i y_i \leq A$$

$$y_i \geq 0, \; y_i \text{ is an integer}, \; i=1,2,\ldots,m$$

Let us call this "unconstrained" problem as P_1.

Problem P_1 is the familiar knapsack problem. Let $[y_1^*, y_2^*, \ldots, y_m^*]$ be a solution to this problem. This solution may not correspond to a feasible cutting pattern. However it gives an upper limit on the total

value that can be obtained.

The objective of the heuristic algorithm is to find a feasible cutting pattern $[a_1^*, a_2^*, \ldots\ldots, a_m^*]$ such that the total value of the objective function (3.3) is maximized. The first step of the algorithm is to find an upper bound solution $[y_1^*, y_2^*, \ldots\ldots, y_m^*]$ to the auxiliary problem P. A better solution than this can not be obtained. It attempts to find out a feasible cutting pattern $[a_1^*, a_2^*, \ldots\ldots, a_m^*]$ such that a_i^* is closer to y_i^* for each demand rectangle. The number of different demand rectangles a_i^*'s should not exceed their respective demands.

The heuristic algorithm tries to fit the demand rectangles corresponding to the given cutting pattern $[y_1^*, y_2^*, \ldots\ldots, y_m^*]$ in the stock rectangle according to certain policies discussed later in detail. It first arranges the demand rectangles in the decreasing order of preference. Let it be assumed that the demand rectangles have been renumbered such that they are in decreasing order of preference. The heuristic algorithm considers the rectangles for fitting in this order. The algorithm has been designed such that it attempts to fit a demand rectangle, say i^{th}, till the total number of it becomes equal to y_i^*. In other words, y_i^* gives an upper bound on the number of i^{th} demand rectangles that can be considered for fitting in this stock plate by the heuristic algorithm.

The algorithm always attempts to fit the first rectangle if possible. If it can not be done, then other demand rectangles are considered for fitting. A demand rectangle can not be fitted in following two situations:

(a) There is not enough space in the stock plate to fit the demand rectangle.

(b) The total units of the demand rectangle equals it's upper bound value y^*.

In situation (b), it is possible that some of the demand rectangles are not being considered for fitting even though there may be enough space for them in the stock plate. There should be some mechanism for considering such demand rectangles for fitting, otherwise the empty space will go waste. It is taken care of in the second iteration of the algorithm. In this iteration, y^* for each rectangle is relaxed, that is upper bound for each demand rectangle is increased upto it's total demand and the algorithm is continued. Thus the resulting solution $[a_1^*, a_2^*, .., a_m^*]$ will represent a feasible cutting pattern.

3.1.1 Some Concepts:

3.1.1a. Identification of a Stock Plate:

Subscripted variable E will be used to denote the length of edges of a plate. For example E_J denotes the length of the J^{th} edge. The serial number of the edges of a plate are defined as :

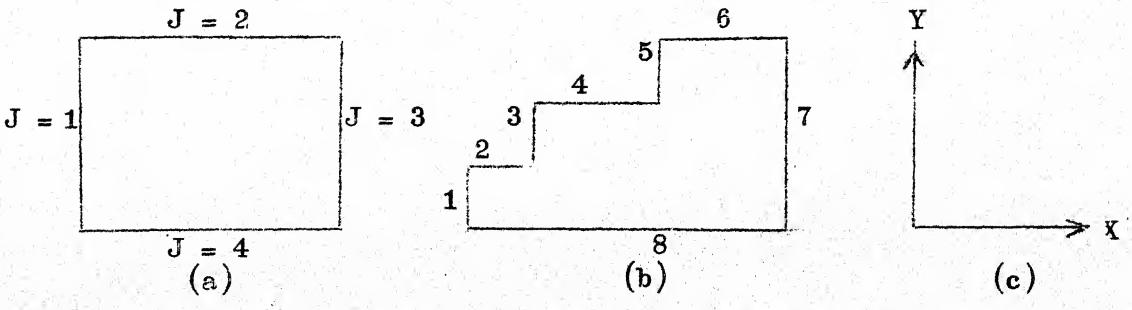


Fig. 3.1: Numbering of Edges of a Stock Plate

According to this definition, odd numbered edges are parallel to the Y - axis and even numbered are parallel to the X - axis. The plate at any stage of cutting can be identified by the values of the subscripted variable E. This system of identifying the plates is not applicable to the following types of situations:

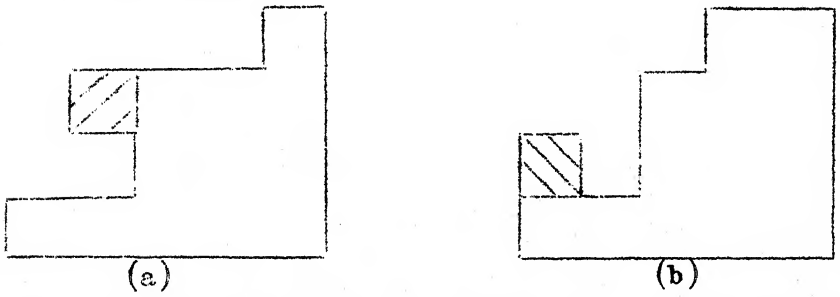


Fig. 3.2: Plates Not Identifiable in Present System

In such a situation, the shaded area shown in figures is treated as waste, so that the resulting plates become identifiable.

A stock rectangle (shown in Fig. 3.1a) will be identified by the following values of the subscripted variable E:

E_1 = Width of the stock rectangle = E_3

E_2 = Length of the stock rectangle= E_4

The widths and lengths of demand rectangles and the stock rectangles are defined such that width $\leq$ length.

3.1.1b. Fitting of a Rectangle in the Stock Plate:

Following information is needed for fitting a demand rectangle in the stock plate and to identify the resulting stock plate:

(i) Subscripted variable E to identify the stock plate.

(ii) The edge J of the stock plate along which the rectangle is to be fitted. The rectangle is always fitted along an odd numbered edge.

(iii) The dimensions of the rectangle to be fitted. Let it be B x L (B and L to be explained later). The edge of the rectangle corresponding to the dimension B is to be placed along J^{th} edge of the stock plate. It is known that there is enough space in the stock plate to fit this rectangle along it's J^{th} edge.

For illustration, consider that a rectangle is to be fitted along 3rd edge of the stock plate shown in Figure (3.3).

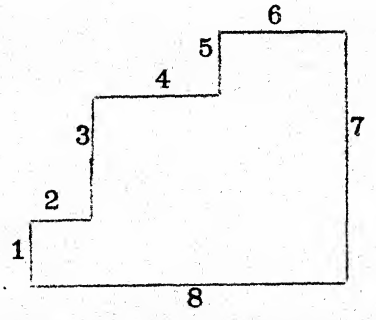


Fig. 3.3: Stock Plate

We have J = 3. One of the situations shown in Fig. (3.4) will occur depending on the dimensions of the stock plate and the rectangle to be fitted. The dotted area shows the position of the rectangle in stock plate, and the hatched area is treated as waste area.

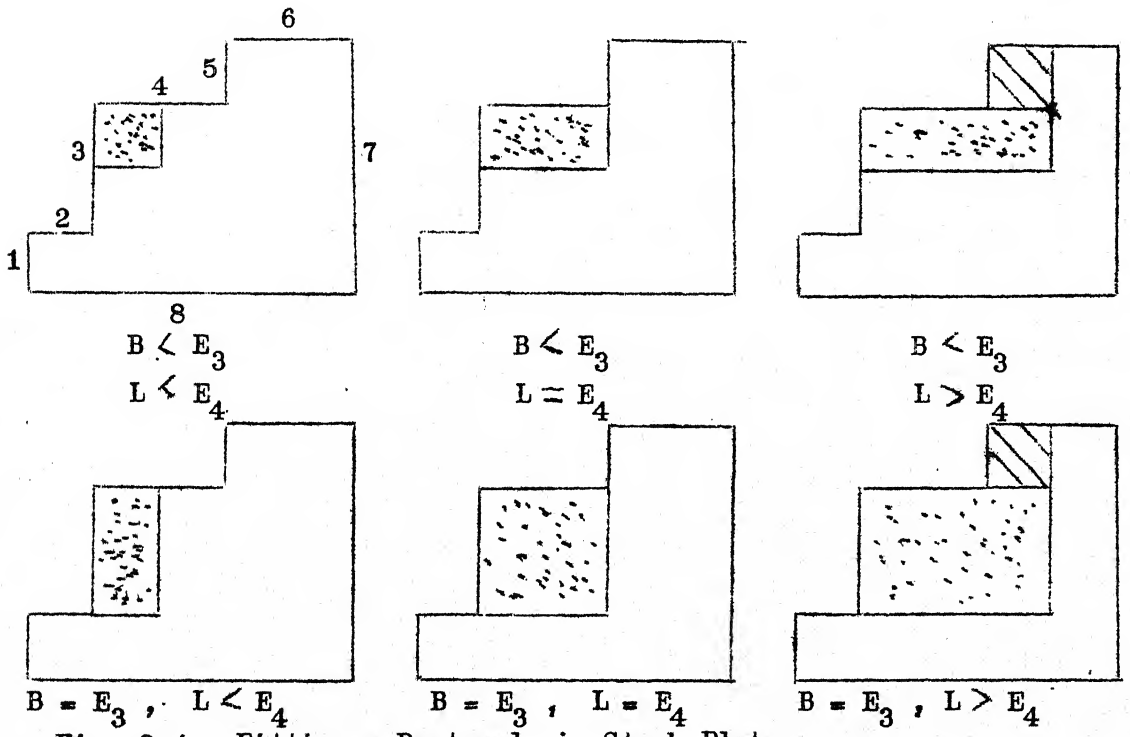


Fig. 3.4: Fitting a Rectangle in Stock Plate

Subscripted variable E can be easily redefined to identify the resulting stock plate.

3.1.1c. Orientation of Demand Rectangle:

The rectangle of dimensions B x L represents either one of the demand rectangles or a group of similar demand rectangles. B and L are calculated depending on the

orientation of the demand rectangles with respect to the stock plate.

Consider that I^{th} demand rectangle is to be fitted along the J^{th} edge of the stock plate. There are two possible orientations for the demand rectangle. The orientation is chosen such that B is nearest possible to E_J (for $J > 1$). Following rules will be followed to choose the orientation. It is known that the demand rectangle can be fitted in at least one of the two orientations.

(i) If $J = 1$, then width of the demand rectangle is placed along first edge of the stock plate. Let w_I denote the width and l_I the length of I^{th} demand rectangle, then $B = w_I$ and $L = l_I$. In case the demand rectangle can not be fitted in this orientation, the other orientation is considered in which $B = l_I$ and $L = w_I$.

(ii) Consider the case when $J > 1$. In this case, the objective is to find out a group of I^{th} demand rectangles which will occupy maximum of J^{th} edge of the stock plate. The group of rectangles will again be a bigger rectangle. It's dimensions are B x L. B and L are calculated from following relationships:

$$B = \max \left\{ \left[\frac{E_J}{w_I}\right] x\, w_I \,,\quad \left[\frac{E_J}{l_I}\right] x\, l_I \right\} \qquad (3.6a)$$

$$L = l_I \text{ if } B = \left[\frac{E_J}{w_I}\right] \times w_I \quad \ldots\ldots \quad (3.6)$$

$$= w_I \text{ otherwise} \quad \ldots\ldots \quad (3.7)$$

B here gives the maximum of J^{th} edge of the stock plate that can be occupied. The square brackets $[\ ]$ here denote the highest integer rounded down.

(iii) In the following situation, the criteria for selecting the orientation is different. Consider the stock plate shown in Fig. (3.5). I^{th} demand rectangle is to be placed along 3rd edge of the stock plate. Two possible orientations for the demand rectangle are shown in Fig. (3.5).

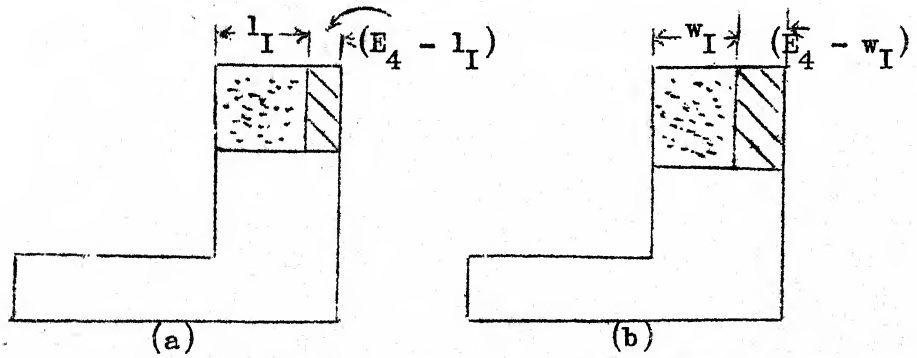


Fig. 3.5: Two Possible Orientations of a Demand Rectangle

If $(E_4 - w_I) <$ minimum of dimensions of demand rectangles, then the hatched area shown in Fig. (3.5b) will go waste. In such a case if $l_I < E_4$, then it is desirable to have the orientation as shown in Fig.(3.5a). If width of the demand rectangle is kept along the 3rd. edge, the waste area will be less. Hence for such cases

we have:

$$B = \left[\frac{E_J}{w_I} \right] \times w_I$$

and $L = l_I$

3.1.1d. Choosing a Proper Rectangle for Fitting:

The rectangles are arranged in decreasing order of preference. They are considered for fitting in this order. I^{th} rectangle is not considered for fitting if $y_I^* = 0$ or the total number of I^{th} demand rectangles fitted is equal to y_I^*. This rule will be followed whenever a demand rectangle is to be chosen for fitting in a stock plate.

3.1.2 Steps of Heuristic Algorithm:

The objective of the heuristic algorithm is to find a feasible cutting pattern $\left[a_1^*, a_2^*, \ldots\ldots, a_m^* \right]$ such that total value of the demand rectangles in the cutting pattern is maximized. The values of different demand rectangles are given. In the linear programming formulation, the values can be found out by following relationship:

$$\left[v_1 , v_2 , \ldots\ldots, v_m \right] = C_B B^{-1}$$

Now the following steps are taken:

(0) Find out the upper bound solution $\left[y_1^*, y_2^*, \ldots, y_m^* \right]$. The upper bound solution can be found by solving the problem P_1. This cutting pattern may be infeasible.

(i) Arrange the demand rectangles in decreasing order of preference. The algorithm arranges the rectangles according to decreasing magnitude of v_i/A_i . The subscripts are reordered such that $v_1/A_1 \geq v_2/A_2 \geq \ldots\ldots \geq v_m/A_m$. The rectangles are considered for fitting in this order. A rectangle is chosen for fitting according to the criteria discussed in (3.1.1d).

(ii) Choose proper demand rectangle I to be fitted on the J^{th} edge of the stock plate. In the beginning, take $J = 1$.

(iii) Find out if I^{th} demand rectangle can be fitted along J^{th} edge. Consider that a rectangle is to be fitted along 3rd edge of the stock plate shown in Fig. (3.3). The demand rectangle can not be fitted with it's width along 3rd edge if:

(a) $w_I > E_3$

(b) $l_I > E_4 + E_6$

Similar conditions can be developed for second orientation also. Go to step (ix) if it can be fitted.

(iv) If it can not be fitted along J^{th} edge, consider other odd numbered edges in the clockwise direction. Go to step (ix) if it can be fitted along any of these edges. In case the rectangle can not be fitted along any of the odd numbered edges, it is concluded that the rectangle can not be fitted in the stock plate at this stage. It does not imply that this

rectangle can not be fitted in the stock plate at other stages afterwards also.

(v) Go to step (vi) if this is the last demand rectangle in the ordered sequence (I = m), other wise choose next demand rectangle for fitting. Go to step (iii) if there is such a demand rectangle in the ordered sequence which can be considered for fitting.

(vi) Go to step (viii) if $y_i^* \neq N_i$, i = 1,2,...,m.

(vii) No more demand rectangles can be cut from the stock plate if it is in rectangular form. Go to step (x) in such a situation. If the stock plate is not in rectangular form, then there is possibility of cutting some more pieces from it by modifying the stock plate as shown in Fig. (3.6):

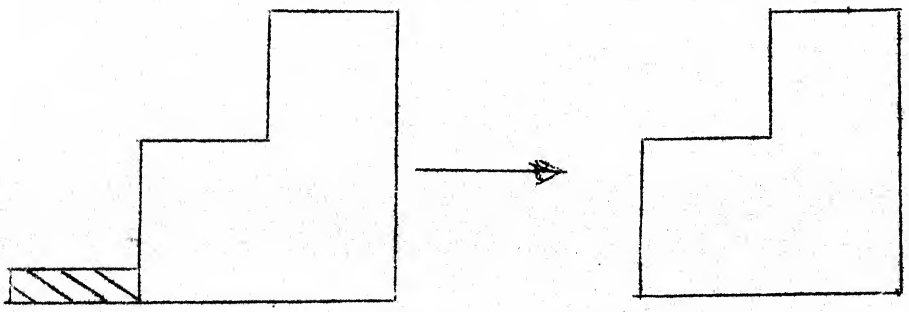


Fig. 3.6: Modifying a Stock Plate

Take J = 1 and choose proper demand rectangle I. Go to step (iii).

(viii) The first iteration of the heuristic algorithm is complete. The total units of a demand rectangle cut so far does not exceed

it's upper bound specified in the cutting pattern found by solving the problem P_1. We go for second iteration if there is possibility of cutting some more demand rectangles from the remaining stock plate, otherwise go to step (x). The upper bounds are relaxed in the second iteration. The algorithm redefines the upper bounds as:

$y_i^* = N_i$, $i = 1, 2, \ldots, m$

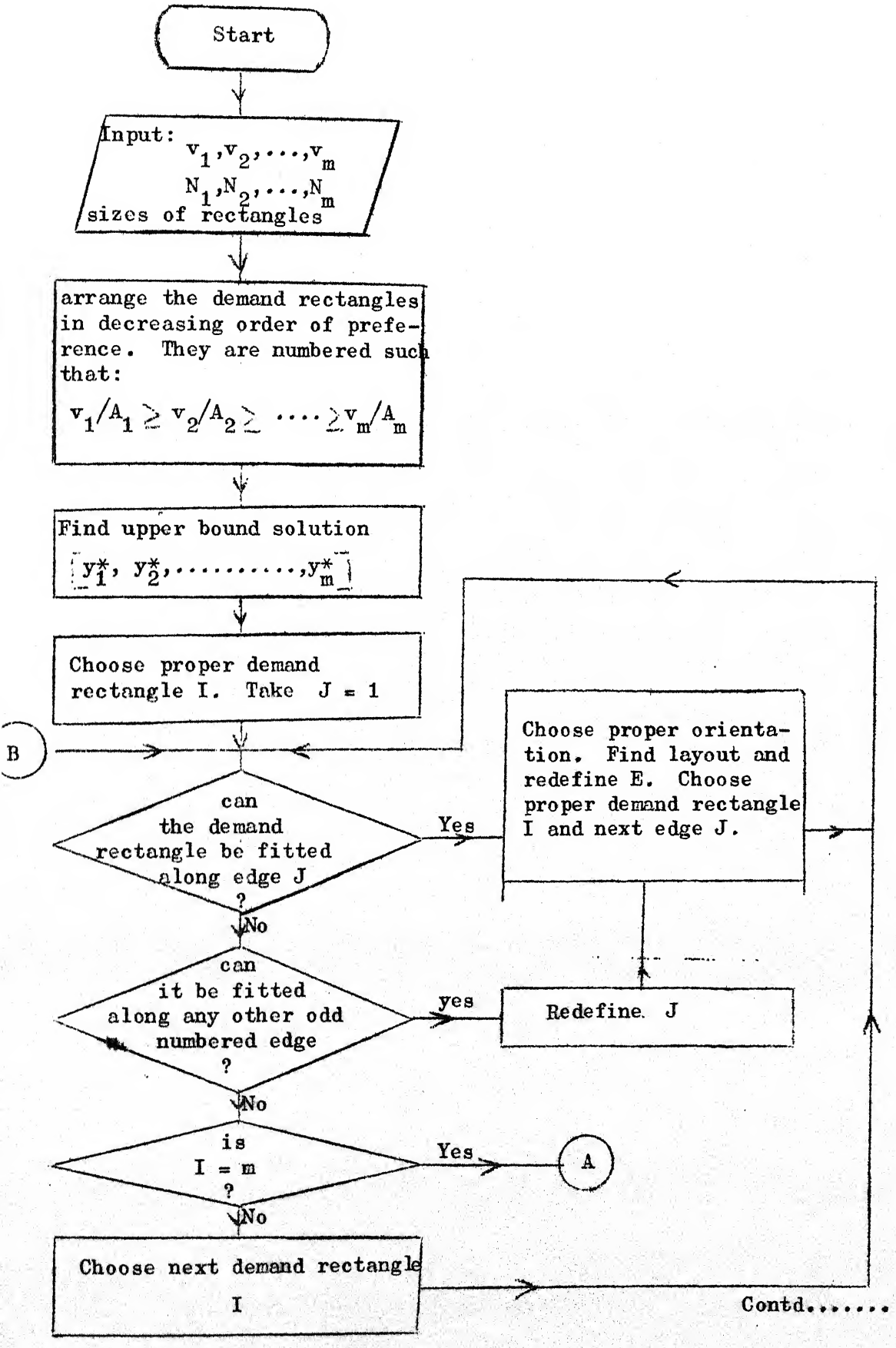
$y_i^* = 0$ if $v_i \leq 0$, $i = 1, 2, \ldots, m$

Take J = 1 and choose proper demand rectangle from the ordered sequence. Go to step (iii).

(ix) Choose proper orientation for the demand rectangle and calculate B and L as discussed before. Find the layout of this rectangle in the stock plate and redefine the subscripted variable E to identify the resulting stock plate.

None of the demand rectangles can be fitted along first edge if E_1 is less than the minimum of dimensions of the demand rectangles. So the stock plate is modified as shown in Fig. (3.6) and the subscripted variable E is redefined. Choose next edge of the stock plate and proper demand rectangle. Go to step (iii).

(x) The algorithm collects all the waste pieces and finds cutting arrangements for them.


Start
Input: $v_1, v_2, \ldots, v_m$ $N_1, N_2, \ldots, N_m$ sizes of rectangles
arrange the demand rectangles in decreasing order of preference. They are numbered such that: $v_1/A_1 \geq v_2/A_2 \geq \ldots \geq v_m/A_m$
Find upper bound solution $[y_1^*, y_2^*, \ldots, y_m^*]$
Choose proper demand rectangle I. Take J = 1
B
can the demand rectangle be fitted along edge J ?
Yes
Choose proper orientation. Find layout and redefine E. Choose proper demand rectangle I and next edge J.
No
can it be fitted along any other odd numbered edge ?
yes
Redefine J
No
is I = m ?
Yes
A
No
Choose next demand rectangle I
Contd.....

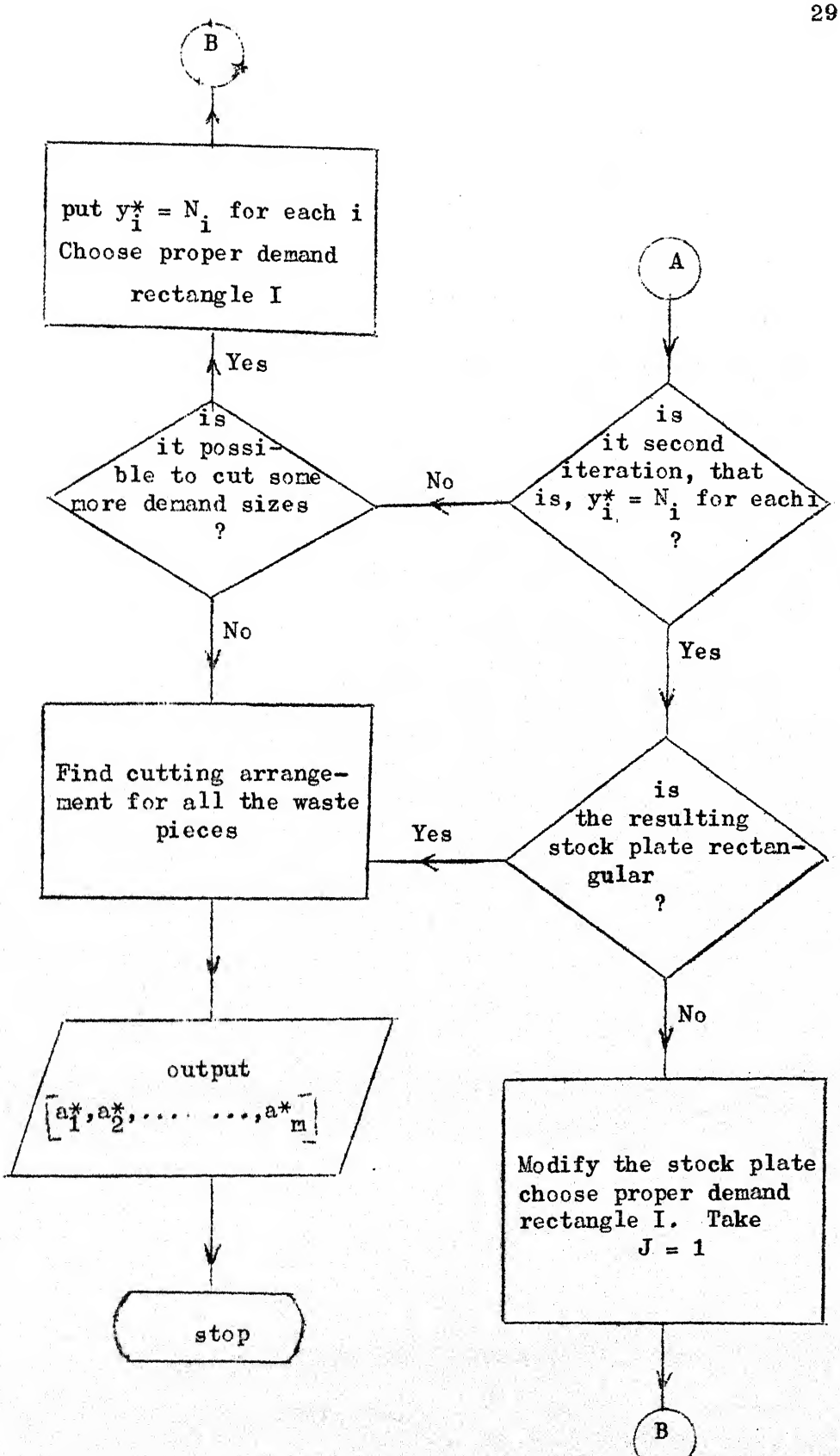


Fig. 3.7 : Flow Chart for Heuristic Algorithm

Finally it finds out the numbers of different demand rectangles that have been fitted. a_i^* denotes the total number of i^{th} demand rectangles for each i .

Flow chart for the algorithm is given in Fig. (3.7).

3.2 Modification of Two Stage Guillotine Cutting Pattern:

A cutting pattern in two stage guillotine cutting sometimes leaves areas which could be further used for cutting demand rectangles. For example, a typical cutting pattern for a strip may be:

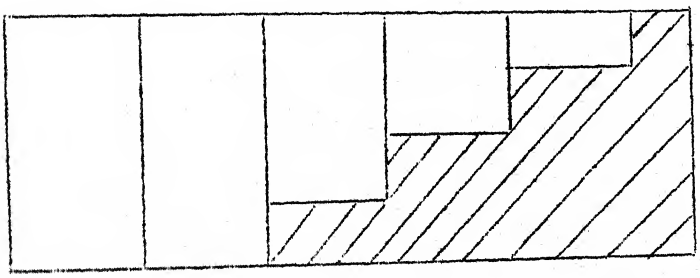


Fig. 3.8 : Two Stage Guillotine Cutting Pattern

The hatched area shown in above figure is treated as waste, while it may be possible to cut some more demand rectangles from this area. It is suggested that the heuristic algorithm developed above can be used to find out if any more demand rectangles can be cut. Following procedure is followed for this:

(1) Define subscripted variable E to identify the hatched area.

(2) Put y_i^* = Total demand for i^{th} rectangle, i = 1,2,....,m.

Start from step (i) of the heuristic algorithm.

The output of the heuristic algorithm will give the number of different demand rectangles that can be cut from the hatched area.

3.3 Solution of Knapsack Problem:

The cutting pattern $[\, y_1^*, y_2^*, \ldots, y_m^* \,]$ in the 0^{th} step of the heuristic algorithm is generated by solving following knapsack problem:

$$\text{Maximize} \quad \sum_{i=1}^{m} v_i y_i \qquad \ldots\ldots \qquad (3.8)$$

$$\text{such that} \quad \sum_{i=1}^{m} A_i y_i \leq A \qquad \ldots\ldots \qquad (3.9)$$

$$y_i \geq 0 \ , \ y_i \text{ is an integer for each } i \qquad (3.10)$$

The variables here have been arranged such that $v_1/A_1 \geq v_2/A_2 \geq \ldots\ldots\ldots \geq v_m/A_m$. This problem could be solved by using dynamic programming or branch and bound approach. Both these methods require a large computer memory. In this thesis, branch and search technique has been used to solve this problem.

In this technique a linear programming problem is solved at each node. Consider this problem at node q.

$$\text{Maximize} \quad \sum_{i=1}^{m} v_i y_i$$

$$\text{such that} \quad \sum_{i=1}^{m} A_i y_i \leq A$$

$$y_i = Z_i^* \ , \text{ for } i \in \{I_1\}$$

$$y_i \geq 0 \ , \quad i = 1,2,\ldots,m$$

where $\{I_1\} = (1, 2, \ldots, q-1)$. This represents the set of indices of assigned variables at the node and Z* represents the values of the assigned variables. A node represents a class of feasible solutions depending on the assigned variables. The branches emanating from a node divide this class into subclasses which are mutually exclusive but exhaustive. For each branch, we have one or more additionally assigned variables along with those assigned at the node from which the branches are emanating . The solution to the linear programming problem is:

$$y_i = Z_i^* \ , \ i = 1,2,\ldots q-1$$

$$y_q = (A - \sum_{i=1}^{q-1} A_i \ Z_i^*) \ / \ A_q$$

$$y_j = 0 \ , \ j = q+1, \ q+2, \ldots, m$$

The value of objective function corresponding to this solution gives an upper bound on the class of feasible solutions represented by this node.

3.3.1 Branching Rules:

Let $y_q = V$ (a fractional number) be the solution of the linear programming problem at a node from which new branches are to be developed. The total number of branches emanating from this node will be equal to $[V] + 1$, and the value of the additional assigned variable y_q for different branches will be 0, 1, 2,....., $[V]$ respectively. Square brackets $[\ ]$ here denote the highest integer rounded down.

First, a branch will be developed from this node with an additional assignment $y_q = 0$. Linear programming problem is solved for the resulting node. Further branching is continued if the solution at the node is fractional and upper bound exceeds the current lower bound. We back track if the solution is integer or the upper bound is less than the lower bound. A node having integer solution will be called as a terminal node. Detailed steps are given below:

(a) Put $q = 1$. It represents 1st. node. Find solution to the linear programming problem at this node, taking $\{I_1\}$ as an empty set. It will give $y_1 = A/A_1$. Find lower bound to the solution. It becomes $v_1 \lceil A/A_1 \rceil$. Put $Z_1^* = 0$. Z_1^* is the value of assigned variable y_1 for a branch coming out from node 1. Put $q = 2$ and go to step (b).

(b) Find Z_1^* , Z_2^* ,...., Z_{q-1}^*. These are the values of the assigned variables y_1, y_2,....,y_{q-1} at q^{th} node. Find solution to the linear programming problem without integer restriction. It will give ; $y_q = (A - \sum_{i=1}^{q-1} A_i Z_i^*) / A_q$. Go to step (c) if y_q is an integer otherwise go to step (d).

(c) Modify the current lower bound if the value of the objective function corresponding to this integer solution is more than the current lower bound. Go to step (e).

(d) Find value of the objective function corresponding to the fractional solution. This is upper bound for this node.

Go to step (e) if upper bound is less than the current lower bound, otherwise further branching is done from this node. Put $Z^*_q = 0$, $q = q+1$ and go step (b).

(e) Go to step (f) if all the branches from $(q-1)^{th}$ node have been developed. Otherwise put $Z^*_{q-1} = Z^*_{q-1} + 1$ and go to step (b).

(f) Stop if $q = 2$. Otherwise put $q = q - 1$ and go to step (e).

This method is illustrated by the help of following example:

Example: Maximize $5x_1 + \frac{10}{3}x_2 + 2x_3$

such that $4x_1 + 3x_2 + 2x_3 \leq 9$

The tree diagram obtained for this problem is shown in Fig.(3.9) The details are described below:

1. Node 1 : $y_1 = 2\frac{1}{4}$, U.B. $= 11\frac{1}{4}$, L.B. $= 10$, Branch from node 1. Go to node 2.

2. Node 2 : $Z^*_1 = 0$, $y_2 = 3$, it is terminal node. Value of objective function $= 10 =$ current lower bound. Back track.

3. Node 2 : $Z^*_1 = 1$, $y_2 = 5/3$, U.B. $= 10\frac{5}{9}$. U.B. $>$ current lower bound, so branching is continued. Go to node 3.

4. Node 3 : $Z^*_1 = 1$, $Z^*_2 = 0$, $y_3 = 5/2$, U.B. $= 10$. U.B. $=$ current L.B. so back track.

5. Node 3 : $Z^*_1 = 1$, $Z^*_2 = 1$, $y_3 = 1$. It is terminal node. Value of objective function $= 10\frac{1}{3}$, which is more than current L.B. so put L.B. $= 10\frac{1}{3}$. Back track.

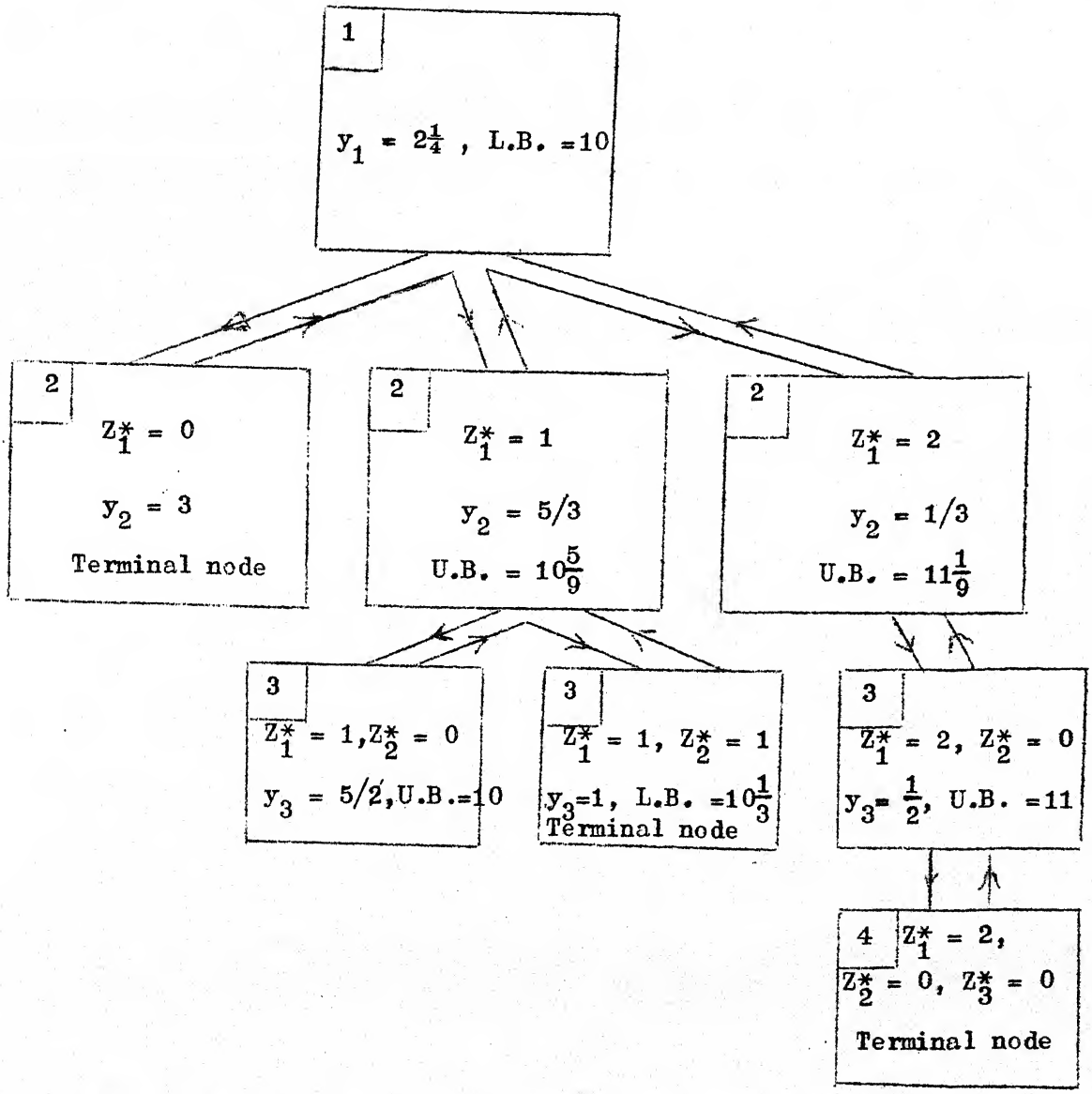


Fig. 3.9: Tree Diagram for Branch and Search Example.

6. : Back track, because all the branches from node 2 have been explored.

7. Node 2 : $Z_1^* = 2$, $y_2 = 1/3$, U.B. = $11\frac{1}{9}$ > current lower bound. Continue branching.

8. Node 3 : $Z_1^* = 2$, $Z_2^* = 0$, $y_3 = 1/2$, U.B. = 11 > current lower bound. Continue branching.

9. Node 4 : $Z_1^* = 2$, $Z_2^* = 0$, $Z_3^* = 0$, it is terminal node. Value of objective function = 10 = current L.B. , back track.

10. : Back track, because all branches from node 2 have been explored.

11. : All branches from node 1 have been explored. Stop because q = 2.

The final solution is the one corresponding to current lower bound. It gives : $y_1^* = 1$, $y_2^* = 1$, $y_3^* = 1$.

3.4 Steps of the Final Algorithm for Solving the Cutting Stock Problem:

These steps are shown in Fig. (3.10) . The symbols used in the flow chart are:

C_B = Cost vector for the activities in the basis.

N = Requirement vector consisting of demands of different rectangles $[N_1, N_2, \ldots, N_m]$.

X_B = Solution vector consisting of m elements. It gives the number of times different activities in the basis are to be repeated.

C = Cost vector consisting of n elements when n is the total number of activities.

Other symbols used are the same as explained before.

The output of the algorithm is:

(i) Basis matrix B.

(ii) Solution vector X_B.

The columns of the basis matrix B give different cutting patterns. There will be total m cutting patterns.

Some of the elements of X_B may be fractional. These numbers are rounded to nearest lower integers. Due to this, the demands for some of the rectangles may be unfilled. So some new cutting patterns are generated to satisfy the demands of these rectangles. The heuristic algorithm for column generation can be used to generate these cutting patterns. Following steps are taken for this:

(i) Put Y_i^* = unfilled demand for i^{th} rectangle for each i. This is taken as an upper bound solution.

(ii) Define value of a demand rectangle equal to it's area,

$$v_i = A_i \ , \ i = 1,2,\ldots,m$$

(iii) Start from first step of the heuristic algorithm. Do not go for second iteration of the algorithm. Find feasible cutting pattern $\left[a_1^*, a_2^*, \ldots, a_m^*\right]$.

(iv) Redefine the upper bound solution:

$$y_{i\,(new)}^* = y_{i\,(old)}^* - a_i^* \ , \ i = 1,2,\ldots,m$$

Stop if $y_i^* = 0$ for each i, otherwise go to step (iii).

Thus, this algorithm will give different cutting patterns and number of times each of them is to be used to fill the order for demand rectangles. The unused areas of the stock rectangles are considered as waste.

Some randomly generated problems have been solved by this method and the results are reported in chapter 4.

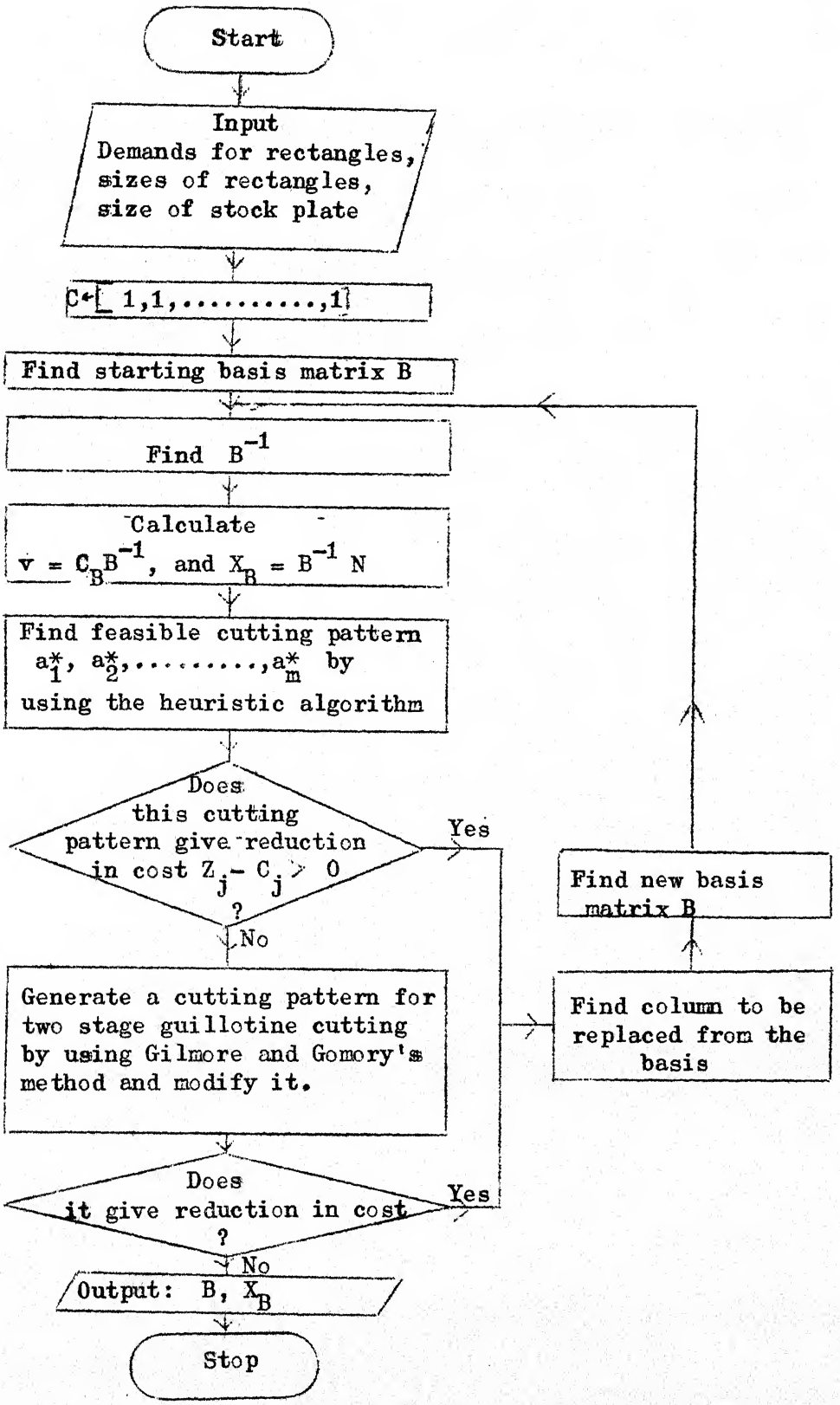


Fig. 3.10: Flow Chart for Final Algorithm

CHAPTER 4

RESULTS AND DISCUSSION

4.1 Heuristic Algorithm for Generating Cutting Patterns:

The objective of this algorithm is to generate a cutting pattern having maximum value. The values and demands of different demand rectangles are given. The number of different demand rectangles in the chosen cutting pattern should not exceed their respective demands.

The first step of the algorithm is to find an upper bound solution $\left[y_1^*, y_2^*, \ldots, y_m^*\right]$. Then, it obtains a feasible cutting pattern $\left[a_1^*, a_2^*, \ldots\ldots a_m^*\right]$ from this such that a_i^* is close to y_i^* for each demand rectangle.

To test this algorithm, a zero waste cutting pattern for a stock plate (Fig. 4.1) was chosen. The value of a demand rectangle is taken equal to it's area. It's demand is taken equal to it's number in the zero waste cutting pattern. These are shown in table(4.1). The size of the stock plate is 14 x 10.

The maximum value of a cutting pattern here is equal to the area of the stock plate. So the zero waste cutting pattern can be taken as the upper bound solution.

TABLE 4.1

Zero Waste Cutting Pattern

Demand Rectangle	Width	Length	Value	Demand	Upper bound solution	Solution by Heuristic Algorithm
R_1	1.0	3.0	3.0	5	5	5
R_2	1.0	2.5	2.5	3	3	3
R_3	1.0	1.5	1.5	1	1	1
R_4	1.0	1.0	1.0	2	2	0
R_5	1.0	2.0	2.0	3	3	3
R_6	1.0	4.0	4.0	2	2	2
R_7	1.5	2.0	3.0	3	3	3
R_8	1.5	3.0	4.5	1	1	1
R_9	2.0	3.0	6.0	5	5	5
R_{10}	2.0	5.0	10.0	1	1	1
R_{11}	2.0	4.0	8.0	1	1	1
R_{12}	2.0	2.5	5.0	3	3	3
R_{13}	2.0	2.0	4.0	3	3	3
R_{14}	3.0	3.5	10.5	1	1	1
R_{15}	0.5	2.0	1.0	1	1	1

The cutting pattern obtained by the heuristic algorithm is shown in column 7 of the table(4.1). Corresponding layout is shown in Fig.(4.2). The total value of this cutting pattern is 138 while the upper bound value is 140. The cutting pattern generated here is near the global

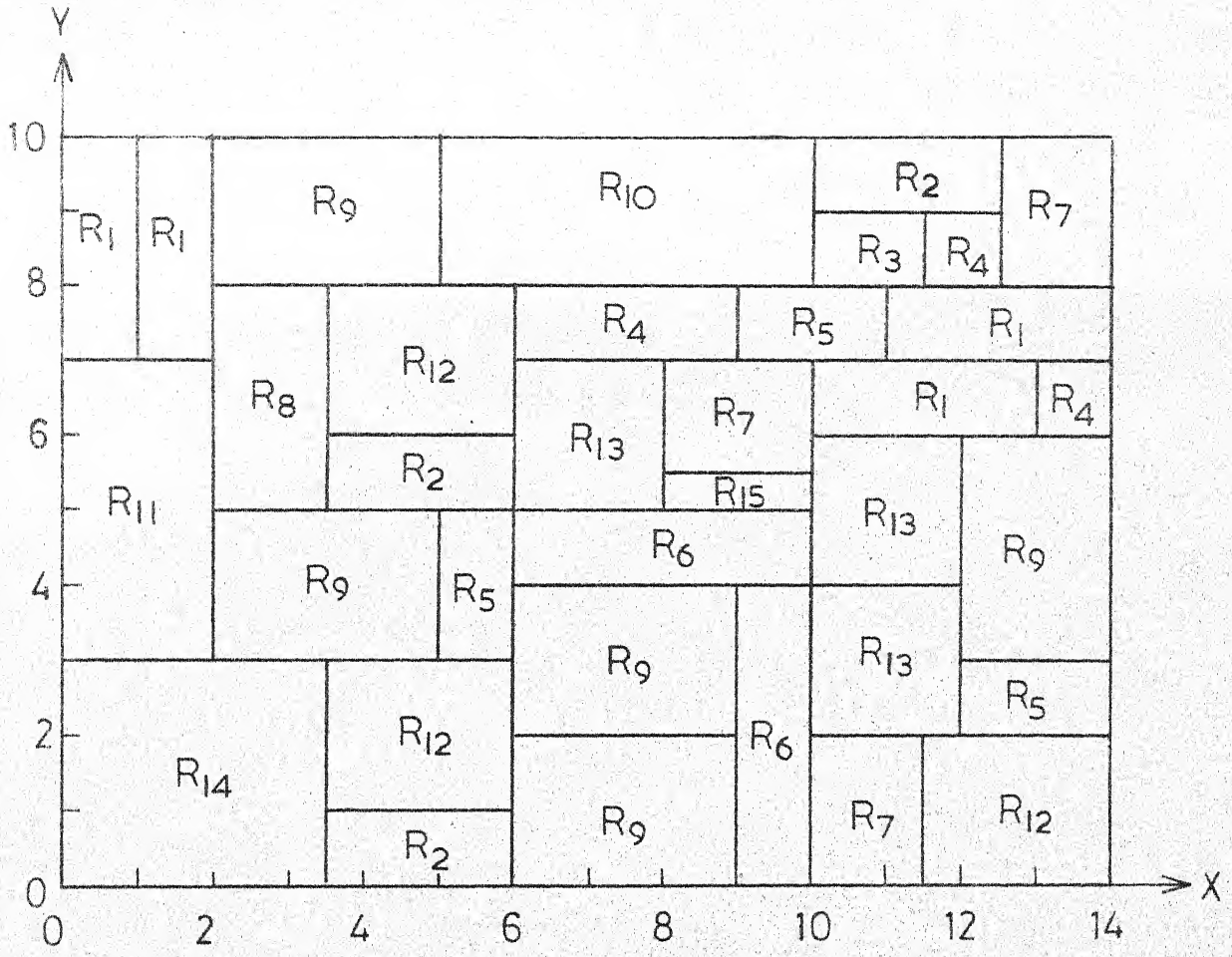


FIG. 4.1 ZERO WASTE CUTTING PATTERN.

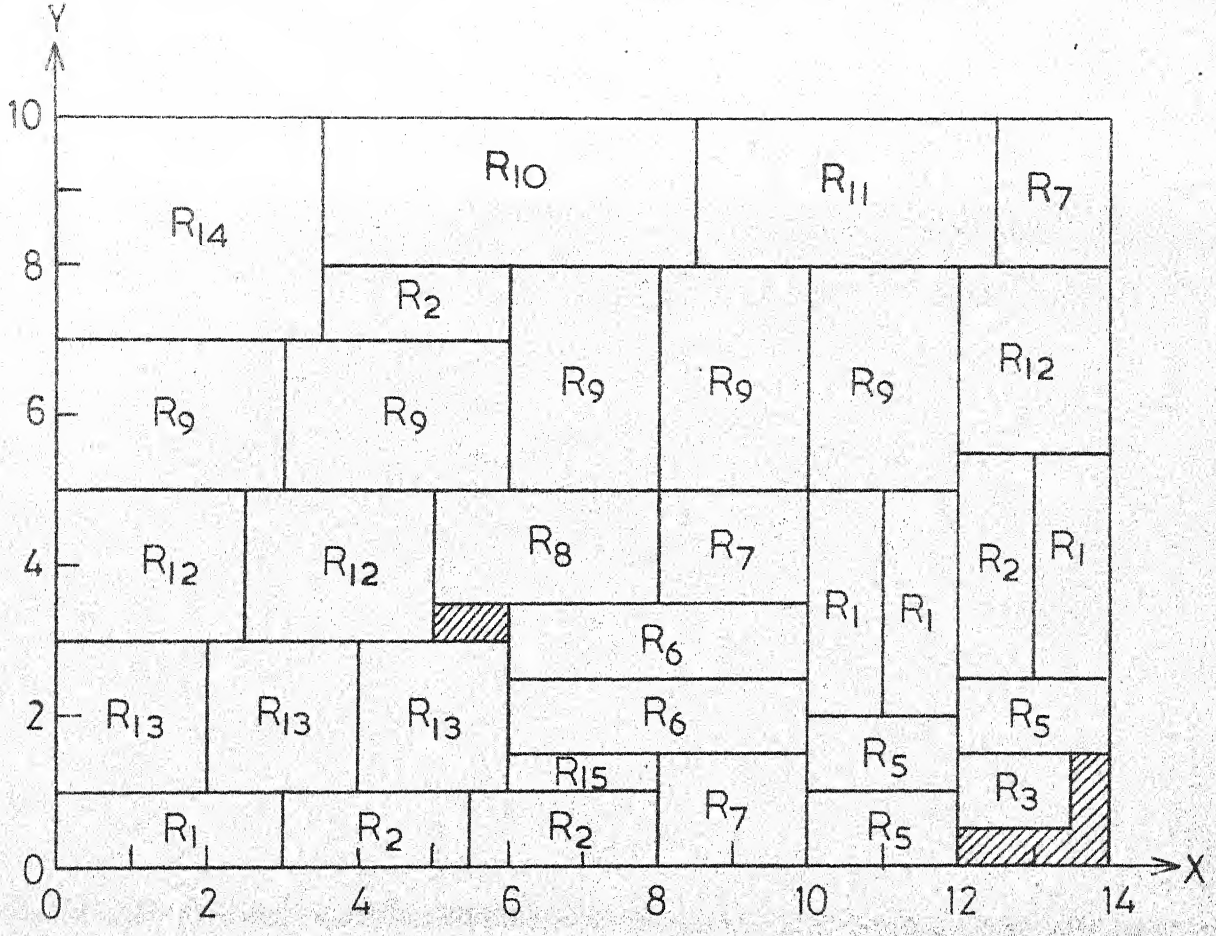


FIG. 4.2 CUTTING PATTERN BY HEURISTIC ALGORITHM.

optimal. The scrap loss of 1.43% is difficult to explain. Since the proposed method is heuristic, the results are quite satisfactory from practical consideration. The results may not be so good for all the problems, but in general it will give good results. It is interesting to note here that time taken for generating this cutting pattern is less than 2 seconds on IBM 7044.

4.2 Algorithm for Solving Cutting Stock Problem:

The objective of this algorithm is to find a cutting arrangement to cut the demand rectangles from the stock rectangles such that the total number of stock rectangles required is minimized, that is, the wastage is minimized.

This algorithm has been tested on some problems reported below. The data for the problems have been generated randomly with uniform distributions. A comparison has been made with the results obtained for the same problems by two stage guillotine cutting method of Gilmore and Gomory.

The data and the results for the problems are shown in tables 4.2, 4.3 and 4.4.

The scrap losses by the heuristic algorithm are smaller in first two problems and same in the third one as compared to the losses by Gilmore and Gomory. It is claimed that the scrap losses by the proposed method will in no case be more. The reason is that this algorithm

TABLE 4.2 - Problem 1

A. Dimensions of Stock Rectangle = 63 x 71

B. Data for the Problem:

Sl. No.	Width	Length	Demand
1	27	33	107
2	3	11	403
3	18	23	349
4	24	24	473
5	2	11	518
6	13	29	691
7	1	6	101
8	7	17	523

C. Results for Problem 1:

	Heuristic	Gilmore and Gomory
Wastage	6.14%	9.78%
Computer Time IBM 7044	2 mts. 11 sec.	19 sec.

TABLE 4.3 - Problem 2:

A. Dimensions of Stock Rectangle = 100.5 x 208.7

B. Data for Problem 2:

Sl.No.	Width	Length	Demand
1	13.5	50.6	2034
2	50.6	85.7	8317
3	63.0	86.5	1602
4	6.3	67.7	4756
5	27.2	68.4	4121
6	53.0	60.3	1923
7	47.0	76.7	4449
8	40.1	86.2	9392
9	19.7	28.6	6313
10	58.1	92.7	3343
11	5.7	76.0	3242
12	35.9	94.3	9364
13	23.5	30.2	7008
14	60.2	88.2	7773
15	24.6	93.2	3565

C. Results for Problem 2:

	Heuristic	Gilmore and Gomory
Wastage	6.1%	8.93%
Computer Time IBM 7044	4 mts. 57 sec.	4 mts. 48 sec.

TABLE 4.4 - Problem 3

A. Dimensions of Stock Plate = 164.8 x 281

B. Data for the Problem:

Sl.No.	Width	Length	Demand
1	46.2	79	2226
2	80.3	55.6	5455
3	20.6	69.1	2704

C. Results for Problem 3:

	Heuristic	Gilmore and Gomory
Wastage	4.16%	4.16%
Computer Time IBM 7044	2 sec.	1 sec.

takes into account all the cutting patterns permitted in two stage guillotine cutting. Not only this, these cutting patterns are further modified before entering into the basis.

In problems 1 and 2, the scrap losses are about 6% by the proposed method and 9% by Gilmore and Gomory's approach. Thus there is reasonable improvement in the scrap saving. However this saving is achieved at the cost of more computer time. The results of problem 2 are interesting, where the computer time taken by both the algorithms are almost same. It is difficult to explain this behaviour.

The computer time taken by this algorithm will depend on the quality of the columns generated. The number of columns generated is going to be very large in case the individual columns generated are of poor quality, and hence more computer time . Since the column generation technique is essentially heuristic, it does not guarantee the generation of best columns.

4.3 Conclusion:

From the results, it is obvious that there is going to be considerable saving in scrap losses by using the algorithm developed here. However, it is at the cost of large computer time.

This method will be very useful for the practical life problems where the material to be cut is costly. There, the saving in scrap may more than makeup for the large computer time.

There is need for further research on this method so that the computer time is reduced.

4.4 Comments on Memory Requirements for Branch and Search:

The branch and search technique for solving the knapsack problems needs very less computer memory. It is because:

(a) It requires storing information for very less number of nodes at a time (maximum m+1 nodes, where m is the number of variables in the problem).

(b) The information stored for each node is also very less. While making calculations at a node, the information stored at other nodes can be utilized.

4.5 Suggestions:

4.5.1 Improving the Heuristic Algorithm for Column Generation:

The 0th step of the algorithm finds upper bound solution by solving following knapsack problem:

$$\text{Maximize} \quad \sum_{i=1}^{m} v_i y_i$$

$$\text{such that} \quad \sum_{i=1}^{m} A_i y_i \leq A$$

y_i is a positive integer for each i .

A better upper bound solution could be obtained if the maximum number of each of the demand rectangles that can be fitted in a stock plate is found out. Let U_i be the maximum number of units of i^{th} demand

rectangle that can be fitted, then the knapsack problem for finding the upper bound solution can be modified as

$$\text{Maximize} \quad \sum_{i=1}^{m} v_i y_i$$

$$\text{such that} \quad \sum_{i=1}^{m} A_i y_i \leq A$$

$$\left.\begin{array}{l} y_i \leq \text{minimum } (U_i , N_i) \\ y_i \text{ is a positive integer} \end{array}\right\} \quad i = 1,2,\ldots\ldots,m$$

where N_i is the demand for i^{th} rectangle.

The additional constraints can be very easily taken into account in the branch and search technique. The time taken for finding upper bound solution will reduce, because the total number of nodes generated will be less.

4.5.2 Solution When Demand Pieces Have Irregular Shapes:

Present work is limited to the problems where all the demand pieces are rectangular. In practical life problems, the demand pieces may have various different shapes. One of the possible ways of solving this problem could be by finding rectangular enclosures for the demand shapes. It can be done by the method suggested by M. Adamowicz and Albano[6]. Then, the heuristic algorithm proposed here can be used to find the cutting arrangements for the rectangular enclosures.

4.5.3 Finding a Cutting Pattern Having Not More Than One Unit of Any Demand Rectangle:

An interesting problem could be: A number of demand

rectangles is given. A cutting pattern for a stock plate is to be found out such that the wastage is minimized. The cutting pattern should not have more than one unit of any demand rectangle. Such a problem can be easily solved by the heuristic for column generation. The only difference will be in finding the upper bound solution in 0th step. In this case, the structure of knapsack problem will be the familiar one where each variable can take only two values 0 or 1.

REFERENCES

1. P.C. Gilmore and R.E. Gomory, "A Linear Programming Approach to the Cutting Stock Problem", Opns. Res. 9, 849-859 (1961)

2. P.C. Gilmore and R.E. Gomory, "A Linear Programming Approach to the Cutting Stock Problem, Part II", Opns. Res. 11, 863-888 (1963)

3. P.C. Gilmore and R.E. Gomory, "Multistage Cutting Stock Problem of Two or More Dimensions", Opns. Res. 13, 94-120 (1965)

4. Susan G. Hahn, "On the Optimal Cutting of Defective Sheets", Opns. Res. 16, 1100-1115 (1968).

5. Kurt Eisman, "The Trim Problem", Mn. Sc. 3, 279-284 (1957)

6. Michel Adamowicz and Antonio Albano, "A Two - Stage Solution of the Cutting Stock Problem", Information Processing 71, Vol. 2, 1086 - 1092

7. Gilmore P.C. and Gomory R.E., "Theory of Knapsack Functions", Opns. Res. 17, 1045 - 1074 (1966)

8. Harold Greenberg, "A Branch and Search Algorithm for Knapsack Problem", Mn. Sc. 16, 327-332 (1970)

ME-1974-M-CHA-TWO